Aaron Borrowman

Fr m: Richard Tad Lepman [tad@bieor.com]
S nt: Tuesday, April 13, 2004 3:39 AM
To: tad@bieor.com
Subject: Rules based Value Analyzer

from url: http://www.teradata.com/t/go.aspx/?id=94623

DATA WAREHOUSING
REPORT
A TERADATA PUBLICATION

AUGUST 2003 VOL 5 ISSUE 4

Solve Specific Business Problems With Rules-Based, Data-Driven Applications

By Brian Wasserman, consulting software analyst

Every business is unique. Each has its own set of customer detail data and its own way of performing functions such as profitability calculations. So how can those businesses meet the customized needs of different customers with a single set of source code? With a Rules-Based, Data-Driven application.

Tailored to customer-specific data
A Rules-Based, Data-Driven Application, such as Teradata Value Analyzer (TVA), is a two-pronged solution that can be customized and tailored to fit the needs of any customer site. The "data-driven" component operates against customer-specific data without requiring changes to the core application software. And the "rules-based" application can process any business rule based on specified templates. Together, the applications permit calculations based on user-defined rules, with the resulting calculations driven by the specifics of customer data.

Here's how a Rules-Based, Data-Driven Application works in a financial institution:
Like other businesses, financial institutions maintain diverse types of detail data about both their customers and their financial transactions. Some institutions keep extremely detailed data, while others are far less sophisticated. At the same time, those institutions also perform profitability calculations in an equally different manner. A "rules-based" application provides them with business rule templates for calculating the various facets of profitability, including revenue, expenses, and risk according to their specific business needs. These templates query for information that users must enter to calculate profitability and include options and choices that are based on customer-specific data.

Effective innovation
Why is this such an innovative — yet effective — approach? Because f the flexibility a Rules-Based, Data-Driven Application can create. First, a Rules-Based, Data-Driven software application can meet the customized needs of different customers with a single set of source

code. Just as important, it allows Sales and Professional Services teams — as well as the customers themselves — to reuse applications such as Teradata Value Analyzer and f cus their energies on s lving a customer business problem, not on engineering software.

Pick the right technology
There are many ways to engineer this type of application. The Value Analyzer architecture, for example, includes a Metadata Definition and a SQL Generator to facilitate the Rules-Based, Data-Driven approach. The Metadata Definition is used to define elements of customer data that will be used in profitability calculations. The SQL Generator uses the Metadata Definitions and generates Teradata-optimized SQL macros that process the dynamic, or customer-specific portions of the profitability rules.

Business Examples
Here are just a few examples of how a Rules-Based, Data-Driven application can solve an array of potential business problems:

Financial Industry: *Apply a fixed $10,000 cost of the bank president to all accounts.*
Here, a financial institution has a fixed amount of money ($10,000), representing the cost of the office of the bank president, that it wishes to spread evenly across all accounts. The software will retrieve the total number of accounts contained in the data warehouse, and will add a cost to each account based on its share of the $10,000. The actual cost applied to each account will vary based on the number of accounts defined.

Travel Industry: *Assign a fixed cost to all flights based on the booking channel.*
In this example, an airline has to pay a fixed ($50) cost for flights that are booked through a travel agency, while flights booked through the airline's web site receive a lesser cost ($1). In this example, a flight is represented by a Passenger-Name-Record (PNR), which is analogous to a master account. A flight booking is a PNR event. The *booking* channel is an attribute of a booking, and is identified to the software through the application metadata during the business discovery process.

The software applies two rules: one acts on PNRs where the *booking channel* = "travel agency;" the second acts on PNRs where the *booking channel* = "web site." The SQL Generator software uses the metadata information to retrieve all PNRs with the appropriate *booking channels* for each rule, insuring that the correct charge is applied.

Communications Industry: *Take a Customer Billed Amount from all customers having calling plan "Family 300", and apply to individual phones based on the total number of minutes for all calls made if the total number of calls is greater than 300 minutes.*

In this example, customers are analogous to master accounts, and phones are equivalent to accounts. During the Business Discovery Process, Customer Attributes *Calling Plan* and *Billed Amount* were identified to the software through metadata. Phones contain calling events which are called Call-Detail-Records (CDRs), and the number of minutes per phone call is an attribute of a CDR.

In processing this Rule, the software uses the *Billed Amount* attribute of a customer as the source for the money being allocated. However, the SQL Generator must first find in the data only those customers with the specified *Calling Plan* = "Family 300." Once these customers have been identified, the SQL Generator must find all phones belonging to these customers and aggregate the number of minutes of phone calls (based on the CDRs). Where this total is greater than 300 minutes, a percentage of the customer *Billed Amount* will be applied.